# 23장. 실행 컨텍스트

1. **실행 컨텍스트(execution context)**

   → 자바스크립트의 동작 원리를 담고 있는 핵심 개념

## 23.1 소스코드의 타입

1. ECMAScript 사양은 소스코드를 4가지 타입으로 구분한다.
   - 전역 코드
   - 함수 코드
   - eval 코드
   - 모듈 코드

   → 4가지 타입으로 구분하는 이유

   : **소스코드 타입에 따라 실행 컨텍스트 생성 과정과 관리 내용이 다르기 때문.**

### 1. 전역 코드

- 전역 코드가 평가되면 전역 실행 컨텍스트가 생성

### 2. 함수 코드

- 함수 코드가 평가되면 함수 실행 컨텍스트가 생성

### 3. eval 코드

- eval 코드가 평가되면 eval 실행 컨텍스트가 생성

### 4. 모듈 코드

- 모듈 코드가 평가되면 모듈 실행 컨텍스트가 생성

## 23.2 소스코드의 평가와 실행

1. 모든 소스코드는 실행에 앞서 평가 과정을 거치며 코드를 실행하기 위한 준비를 함.

   ➡️ 자바스크립트 엔진은 소스코드를 “소스코드 평가”, “소스코드의 실행” 으로 나누어 처리.

2. 소스코드 평가 과정에서는 실행 컨텍스트를 생성하고 변수, 함수 등의 선언문만 먼저 실행

   → 생성된 변수나 함수 식별자를 키로 실행 컨텍스트가 관리하는 스코프에 등록

3. 소스코드 평가 과정이 끝나면, 선언문 제외한 소스코드가 순차적으로 실행

   = 런타임 시작

## 23.3 실행 컨텍스트의 역할

### 1. 전역 코드 평가

- 선언문만 먼저 실행
- 즉, 전역 코드의 변수 선언문, 함수 선언문 실행

  → 실행 컨텍스트가 관리하는 전역 스코프에 등록됨

### 2. 전역 코드 실행

- 전역 코드 평가 과정이 끝나면 런타임이 시작되어 전역 코드가 순차적으로 실행
- 전역 변수에 값 할당
- 함수 호출 -> 순차적으로 실행되던 전역 코드의 실행을 일시 중단하고 코드 실행 순서를 변경하여 함수 내부로 진입

### 3. 함수 코드 평가

- 매개 변수, 지역 변수 선언문 실행 → 지역 스코프에 등록

### 4. 함수 코드 실행(again)

- 함수 코드 순차적으로 실행
- 매개변수와 지역 변수에 값 할당됨 -> console.log 메서드 호출
- console.log 메서드를 호출하기 위해 먼저 식별자인 console을 스코프 체인을 통해 검색한다.

  console 식별자는 스코프 체인에 등록되어 있지 않고 전역 객체에 프로퍼티로 존재한다.

  ( =**전역 객체의 프로퍼티가 마치 전역 변수처럼 전역 스코프를 통해 검색 가능해야한다는 것** )

## 23.4 실행 컨텍스트 스택

1. [ 예제 23-03 ]

```
const x = 1;

function foo () {
  const y = 2;

  function bar () {
    const z = 3;
    console.log(x + y + z);
   }
   bar();
 }

 foo(); // 6
```

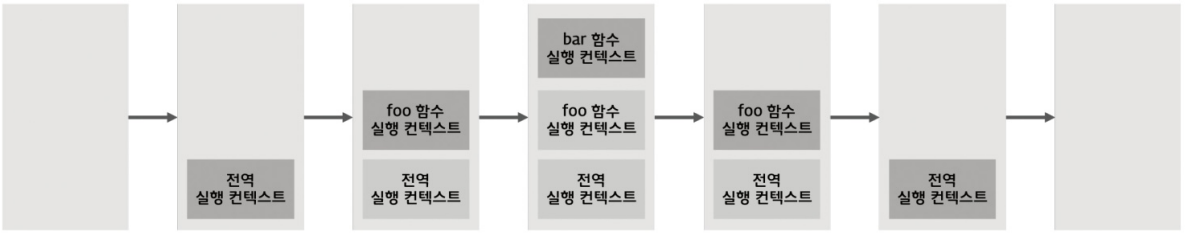


그림 23-5 실행 컨텍스트 스택

- 실행 컨텍스트 스택은 코드의 실행 순서를 관리한다.

  소스코드가 평가되면 실행 컨텍스트가 생성되고 실행 컨텍스트 스택의 최상위에 쌓인다.
- 실행 컨텍스트 스택의 최상위에 존재하는 실행 컨텍스트는 언제나 **현재 실행 중인 코드의 실행 컨텍스트다.**
- 따라서 실행 컨텍스트 스택의 최상위에 존재하는 실행 컨텍스트를 실행 중인 **실행 컨텍스트**라 부른다.

## 23.5 렉시컬 환경

1. 렉시컬 환경은 식별자와 식별자에 바인딩된 값, 그리고 상위 스코프에 대한 참조를 기록하는

   자료구조로 실행 컨텍스트를 구성하는 컴포넌트다.

2. 실행 컨텍스트 스택 : 코드의 실행 순서 관리

   렉시컬 환경 : 스코프와 식별자 관리

3. **환경 레코드**

   : 스코프에 포함된 식별자를 등록하고 등록된 식별자에 바인딩된 값을 관리하는 저장소

4. **외부 렉시컬 환경에 대한 참조**

   : 해당 실행 컨텍스트를 생성한 소스코드를 포함하는 상위 코드의 렉시컬 환경을 가리킴.

## 23.6 실행 컨텍스트의 생성과 식별자 검색 과정

1. [ 예제 23-04 ]

```
var x = 1;
const y = 2;

function foo (a) {
  var x = 3;
  const y = 4;

  function bar (b) {
    const z = 5;
    console.log(a + b + x + y + z);
  }
  bar(10);
 }

 foo(20); // 42
```

- 전역 코드 평가 : 전역 실행 컨텍스트 생성 - 전역 렉시컬 환경 생성 - 전역 환경 레코드 생성 - 객체 환경 레코드 생성 - 선언적 환경 레코드 생성 - this 바인딩 - 외부 렉시컬 환경에 대한 참조 결정
- foo 함수 코드 평가 : 함수 실행 컨텍스트 생성 - 함수 렉시컬 환경 생성 - 함수 환경 레코드 생성 - this 바인딩 - 외부 렉시컬 환경에 대한 참조 결정
- bar 함수 코드 실행 : console 식별자 검색 - log 메서드 검색 - 표현식 a + b + x + y + z의 평가 - console.log 메서드 호출

## 23.7 실행 컨텍스트와 블록 레벨 스코프

1. let, const로 선언한 변수는 모든 코드 블록(함수, if 문, for 문, while 문, try/catch 문 등)을 지역 스

코프로 인정하는 블록 레벨 스코프를 따른다.

2. [ 예제 23-11 ]

```
let x = 1;

if (true) {
    let x = 10;
    console.log(x); // 10
}

console.log(x); // 1
```

- if 문 코드 블록 실행 - 렉시컬 환경 새롭게 생성 - 블록 레벨 스코프 생성
- for 문의 코드 블록이 반복 실행될 때마다 독립적인 렉시컬 환경 생성하여 식별자 값 유지